

# Logistic Regression Model

## Cost Function

Supervise Learning Problem

Training set:  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

m examples  $x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

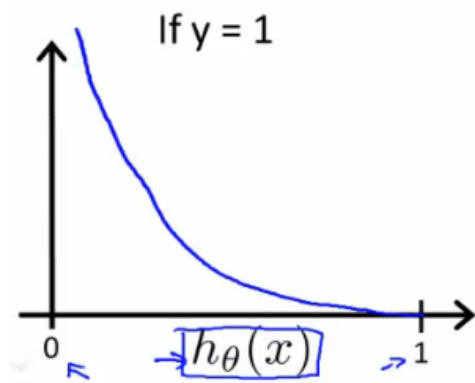
## How to choose parameters $\theta$ ?

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

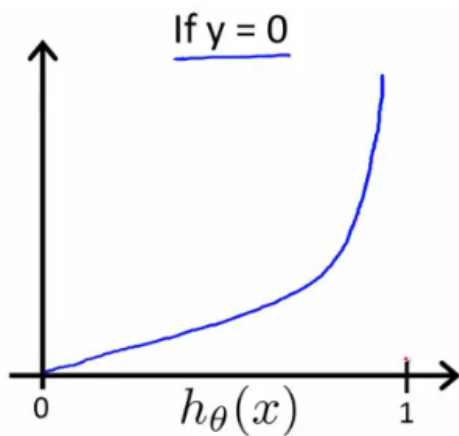
Instead, our cost function for logistic regression looks like:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$
$$\begin{aligned} \text{Cost}(h_{\theta}(x), y) &= -\log(h_{\theta}(x)) & \text{if } y = 1 \\ \text{Cost}(h_{\theta}(x), y) &= -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{aligned}$$

When  $y = 1$ , we get the following plot for  $J(\theta)$  vs  $h_{\theta}(x)$ :



Similarly, when  $y = 0$ , we get the following plot for  $J(\theta)$  vs  $h_{\theta}(x)$ :



$\text{Cost}(h_\theta(x), y) = 0$  if  $h_\theta(x) = y$   
 $\text{Cost}(h_\theta(x), y) \rightarrow \infty$  if  $y = 0$  and  $h_\theta(x) \rightarrow 1$   
 $\text{Cost}(h_\theta(x), y) \rightarrow \infty$  if  $y = 1$  and  $h_\theta(x) \rightarrow 0$

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Note that writing the cost function in this way guarantees that  $J(\theta)$  is convex for logistic regression.

## Simplified Cost Function and Gradient Descent

We can compress our cost function's two conditional cases into one case:

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

Notice that when y is equal to 1, then the second term  $(1 - y) \log(1 - h_\theta(x))$  will be zero and will not affect the result. If y is equal to 0, then the first term  $-y \log(h_\theta(x))$  will be zero and will not affect the result.

We can fully write out our entire cost function as follows:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

A vectorized implementation is:

$$\begin{aligned}
 h &= g(X\theta) \\
 J(\theta) &= \frac{1}{m} \cdot \left( -y^T \log(h) - (1 - y)^T \log(1 - h) \right)
 \end{aligned}$$

## Gradient Descent

Remember that the general form of gradient descent is:

```
Repeat {  
   $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$   
}
```

We can work out the derivative part using calculus to get:

```
Repeat {  
   $\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$   
}
```

Notice that this algorithm is identical to the one we used in linear regression. We still have to simultaneously update all values in theta.

A vectorized implementation is:

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

## Advanced Optimization

"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize  $\theta$  that can be used instead of gradient descent. We suggest that you should not write these more sophisticated algorithms yourself (unless you are an expert in numerical computing) but use the libraries instead, as they're already tested and highly optimized. Octave provides them.

We first need to provide a function that evaluates the following two functions for a given input value  $\theta$ :

$$J(\theta)$$
$$\frac{\partial}{\partial \theta_j} J(\theta)$$

We can write a single function that returns both of these:

```
1 function [jVal, gradient] = costFunction(theta)  
2   jVal = [...code to compute J(theta)...];  
3   gradient = [...code to compute derivative of J(theta)...];  
4 end
```

Then we can use octave's "fminunc()" optimization algorithm along with the "optimset()" function that creates an object containing the options we want to send to "fminunc()". (Note: the value for MaxIter should be an integer, not a character string - errata in the video at 7:30)

```
1 options = optimset('GradObj', 'on', 'MaxIter', 100);  
2 initialTheta = zeros(2,1);  
3 [optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta,  
4     options);
```

We give to the function "fminunc()" our cost function, our initial vector of theta values, and the "options" object that we created beforehand.

## Multiclass Classification: One-vs-all

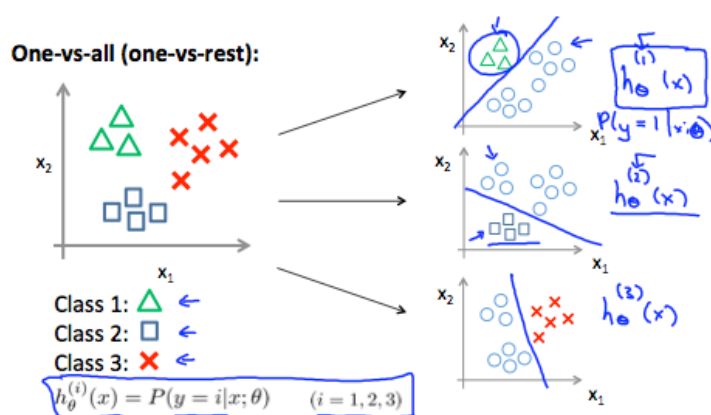
Now we will approach the classification of data when we have more than two categories. Instead of  $y = \{0, 1\}$  we will expand our definition so that  $y = \{0, 1 \dots n\}$ .

Since  $y = \{0, 1 \dots n\}$ , we divide our problem into  $n+1$  (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$$\begin{aligned} y &\in \{0, 1 \dots n\} \\ h_{\theta}^{(0)}(x) &= P(y = 0|x; \theta) \\ h_{\theta}^{(1)}(x) &= P(y = 1|x; \theta) \\ &\dots \\ h_{\theta}^{(n)}(x) &= P(y = n|x; \theta) \\ \text{prediction} &= \max_i (h_{\theta}^{(i)}(x)) \end{aligned}$$

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

The following image shows how one could classify 3 classes:



Train a logistic regression classifier  $h_{\theta}^{(i)}(x)$  for each class  $i$  to predict the probability that  $y = i$ .

On a new input  $x$ , to make a prediction, pick the class  $i$  that maximizes

$$\max_i h_{\theta}^{(i)}(x)$$